Principal Component Transforms in Remote Sensing and Image Processing using Eigenvalues and Eigenvectors.

Linear Algebra

Differential Equations

Remote Sensing for Research

Dr. Bill Capehart
Institute of Atmospheric Sciences
SD School of Mines

Introduction

- Eigenvalues and Eigenvectors are a pair of concepts used in classes from dynamics to image processing.
- This presentation shows an application of these concepts towards image processing using a method called

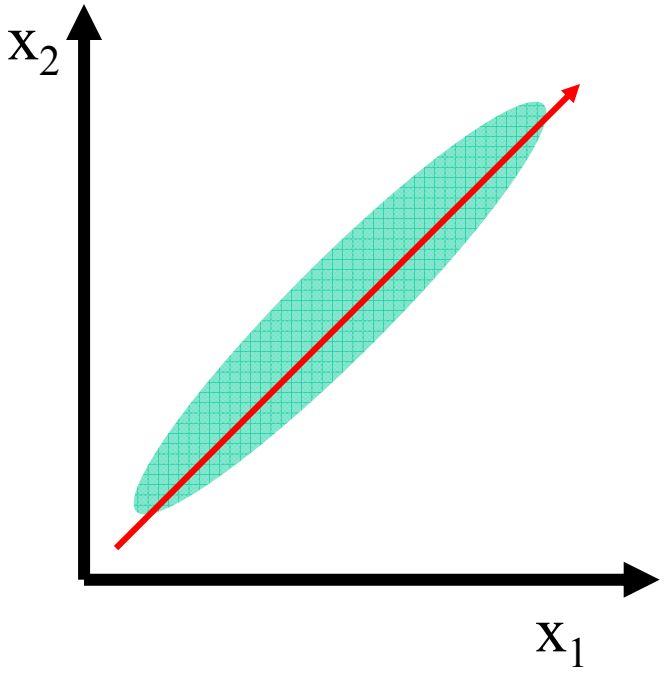
Principal Component Analysis

What is an image transform?

- In image transform is the reorganization of a set of multi-dimensional data (e.g, multispectral imagery) so that key features (such as the relative distance, magnitudes and angles between pixels) are retained in some fashion.
- This is best described by rotating a set of axes:

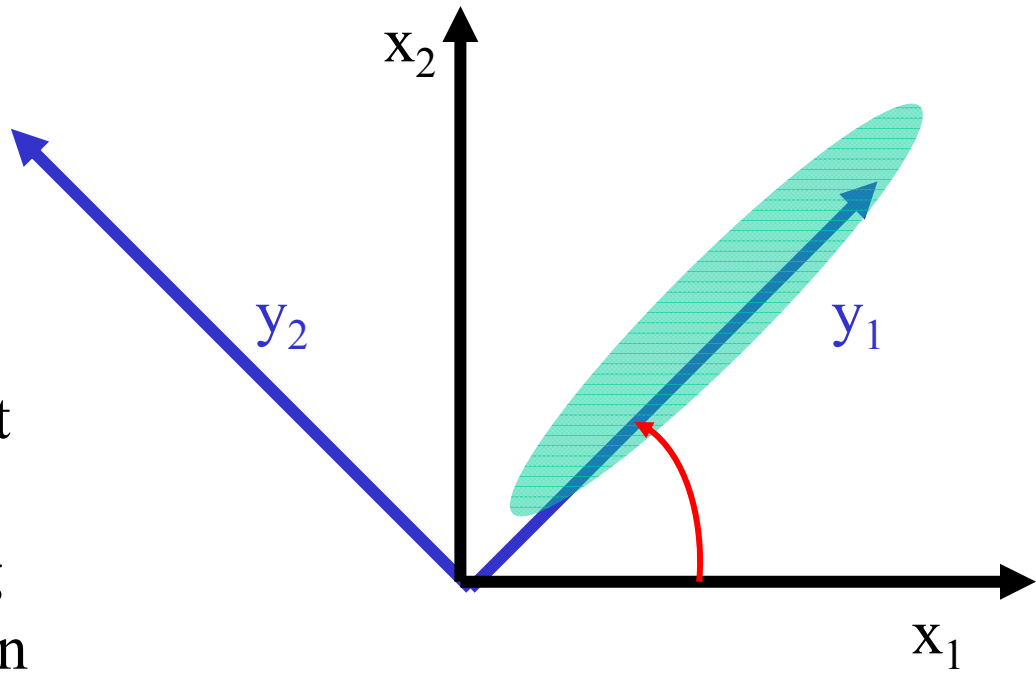
Axis Rotation

- Consider this example:
  - This features a highly correlated pair of image bands (or two layers of any dataset), resented in a scatterplot of x_1-x_2 state space.
  - One may argue that they each may carry significant information. However, most of the information appears to move along a single axis in x_1-x_2 state space.
  - Thus, in a sense, the two layers of information used to create this scatterplot are mutually redundant to a large degree.
  - Fusing both layers into a single layer of information may be an effective solution for such datasets.
  - But how do we do this in a matter that is quantitatively sound and manageable?



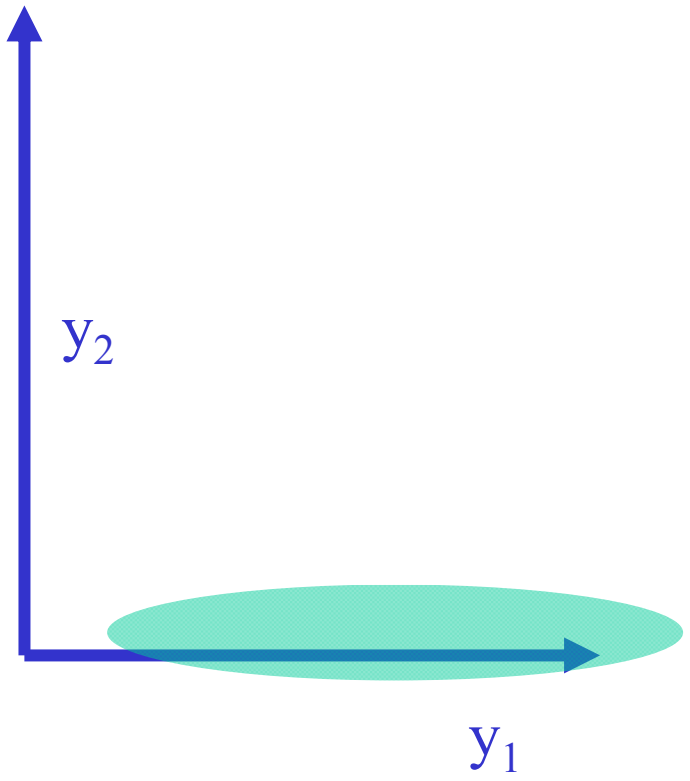
Axis Rotation

- Let's then modify this arrangement of data:
  - While keeping the axial orientation orthogonal (at 90°) to one another, we rotate these two axes so that most of the datasets information is carried along one axis (which we will then refer to as a "component" or composite parameter, y_1, while the remainder of the signal is represented by it's orthogonal companion, y_2.



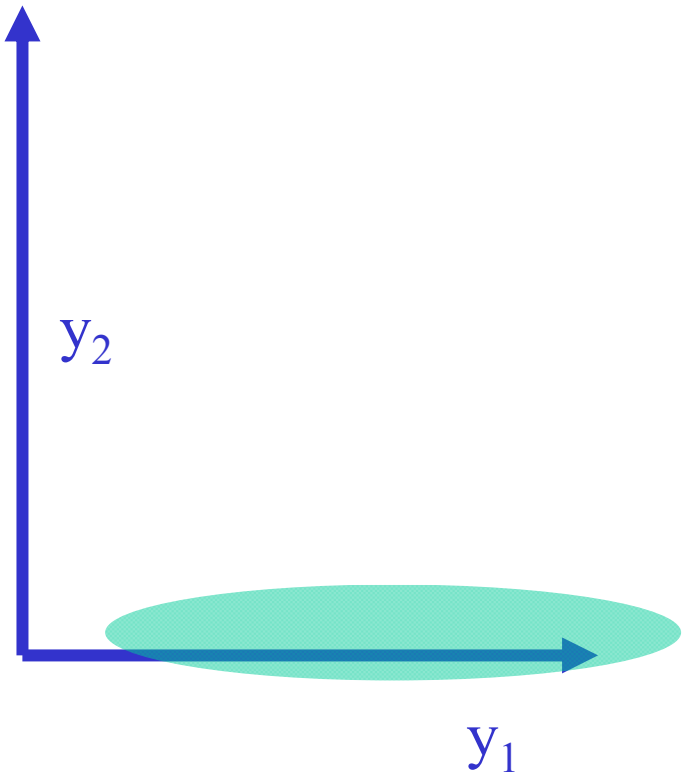
Axis Rotation

- The Specifics
  - Numerically and Statistically, we have done the following:
    - We have greatly reduced the net correlation between our data dimensions or layers (we represent these as satellite bands but these could easily be precipitation, temperature or stream flow).
    - We have retained the original *relative* structure of the dataset.



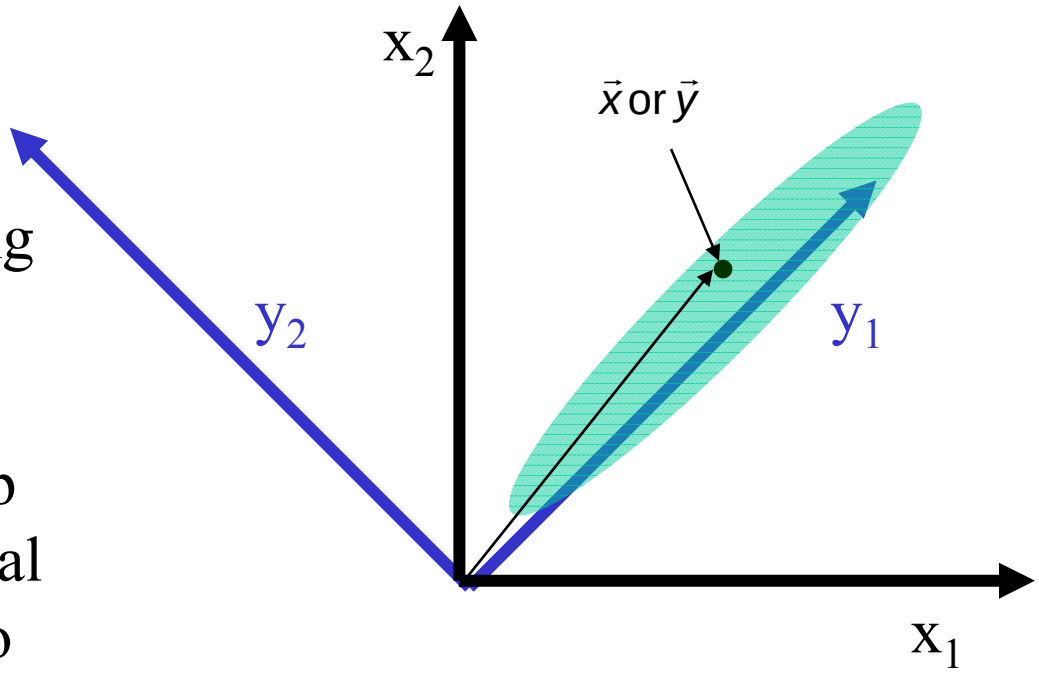
Axis Rotation

- The Consequences
  - We now have a dataset that has one primary attribute: the distance in the direction of y_1 and a secondary attribute of lesser import (the length of y_2).
  - This composite metric allows us to analyze large multidimensional dataset with potentially fewer *significant* attributes.
    - Think Hyperspectral Images: The ones with 200+ Channels of Information!



Axis Rotation

- Mathematically how is this done?
- Here, we are going to use a matrix solving approach using the concept of Eigenvalues and Eigenvectors.
- But FIRST, we need to set up our problem in a mathematical environment more friendly to a Linear Algebra problem solving environment.

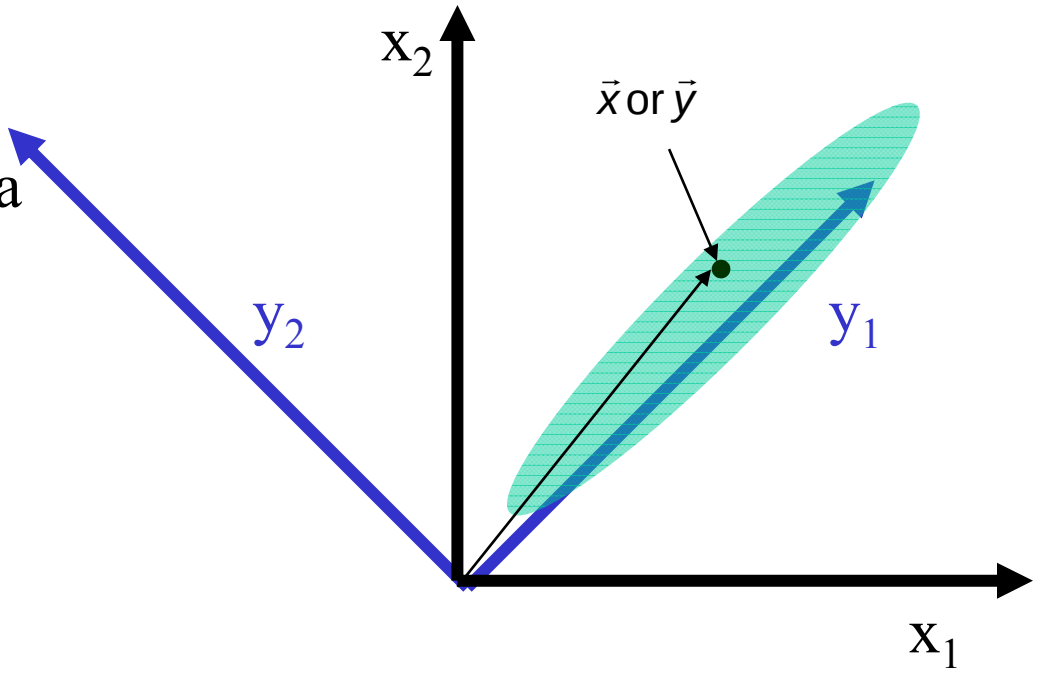


Axis Rotation

- Setting the Stage for Our Solution
  - Let's first arrange our data into a matrix/vector format.
  - Any data point can be expressed (in 2-D space) as

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ or } \vec{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

  - Higher order datasets are expressed similarly…



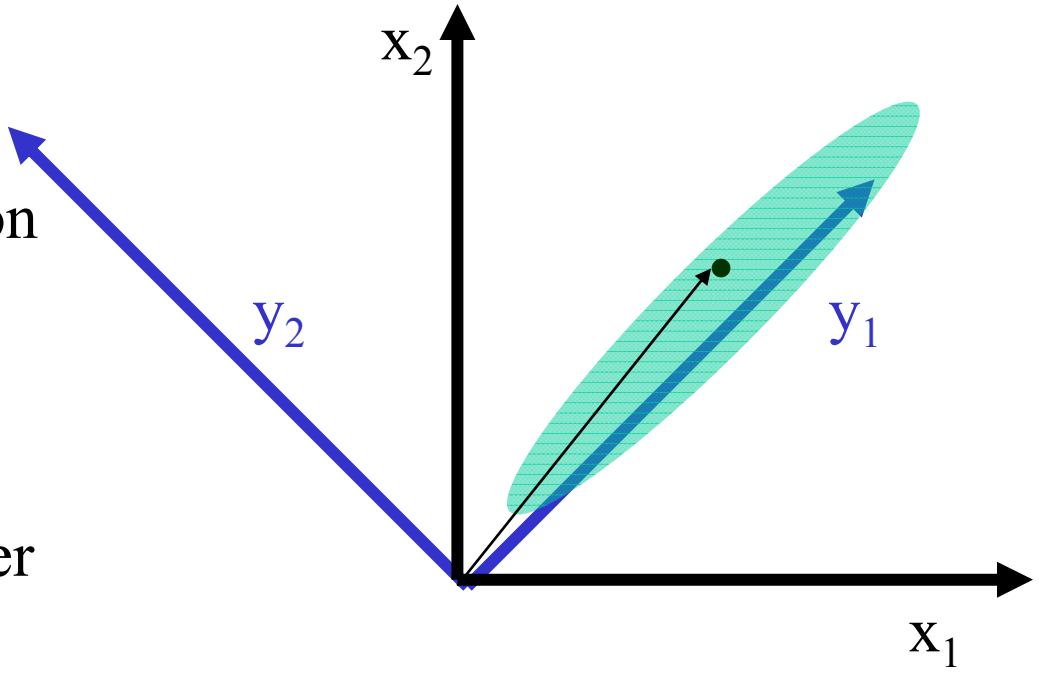
$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Axis Rotation to Vector/Matrix Transforms

- How is this done?
  - To move between the x and y reference frames, we apply a transformation matrix "**G**" so that

$$\vec{y} = \mathbf{G}\,\vec{x}$$

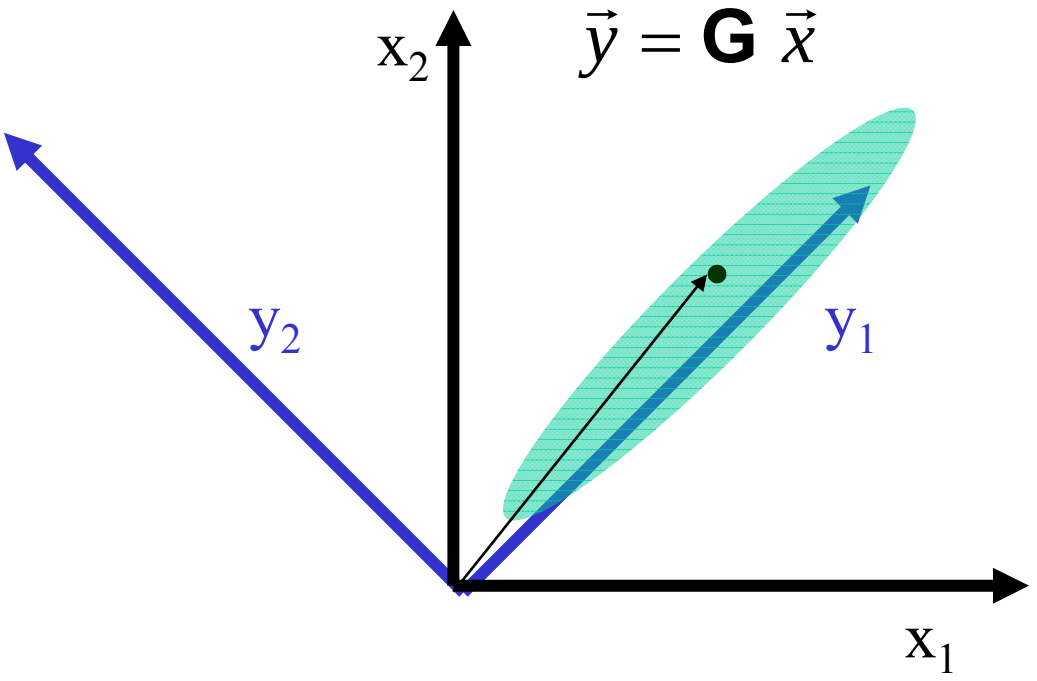
  - **G** has the same dimensions as the number of dimensions in the system.



$$\vec{y} = \mathbf{G}\,\vec{x} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 g_{11} + x_2 g_{12} \\ x_2 g_{21} + x_2 g_{21} \end{bmatrix}$$

Vector/Matrix Transforms

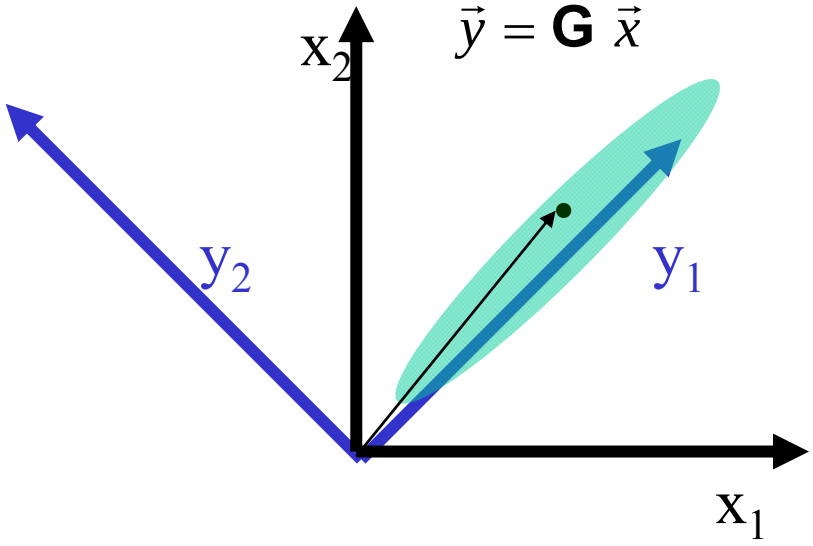
- Given this, one must now determine how generate an appropriate transformation matrix.
- By what criterion to you want to rotate an axis?
  - Reduce the overall correlation to facilitate cross-dimensional data analysis? (Principal Components Transform/Analysis)
  - Minimize the noise in an image (MNF Transform)
  - Highlight a standardized set of multidimensional properties (Tasseled Cap Transform)



- We shall focus here on the PCA

Principal Components Transform

- Designed to rotate a set of axes so that the resulting overall correlation, $\mathbf{\Sigma_y}$, between image bands are minimized.
- Using a highly correlated dataset as an example:



$$\vec{y} = \mathbf{G}\ \vec{x}$$

$$\Sigma_x = \begin{bmatrix} \sigma_{x_1}^2 = 1.90 & \sigma_{x_1x_2} = 1.10 \\ \sigma_{x_2x_1} = 1.10 & \sigma_{x_2}^2 = 1.10 \end{bmatrix}$$

$$\Sigma_y = \begin{bmatrix} \sigma_{y_1}^2 & 0 \\ 0 & \sigma_{y_2}^2 \end{bmatrix}$$

$$\Sigma_y = \mathbf{G}\Sigma_x\mathbf{G}^\mathbf{T}$$

- So the, HOW DO WE GET **G** ?!?!?

$$\vec{y} = \mathbf{G}\ \vec{x}$$

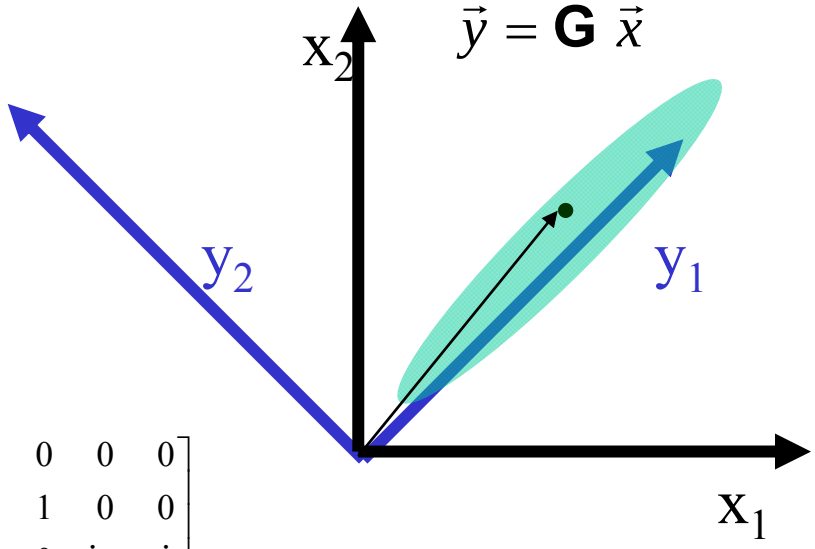
$$\Sigma_x = \frac{1}{N-1}\sum_{n=1}^{N}\left[\vec{x}_n - \vec{\bar{x}}\right]\left[\vec{x}_n - \vec{\bar{x}}\right]^T$$

$$\Sigma_y = \frac{1}{N-1}\sum_{n=1}^{N}\left[\mathbf{G}\vec{x}_n - \mathbf{G}\vec{\bar{x}}\right]\left[\mathbf{G}\vec{x}_n - \mathbf{G}\vec{\bar{x}}\right]^T$$

$$\Sigma_y = \mathbf{G}\Sigma_x\mathbf{G}^\mathbf{T}$$

Diagonalizing a Matrix

- Notice that $\mathbf{\Sigma}_y$ is the "diagonal" matrix of $\mathbf{\Sigma}_x$. (no covariance)
- Moreover, **G** must yield an orthogonal transform. $\mathbf{GG^T = I}$
- (diagonal matrices are also orthogonal)
- We want a solution for $\mathbf{\Sigma}_y$ that is both diagonal (w/o covariance), and yields an orthogonal transformation matrix "**G**"



$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

$$\vec{y} = \mathbf{G}\,\vec{x}$$

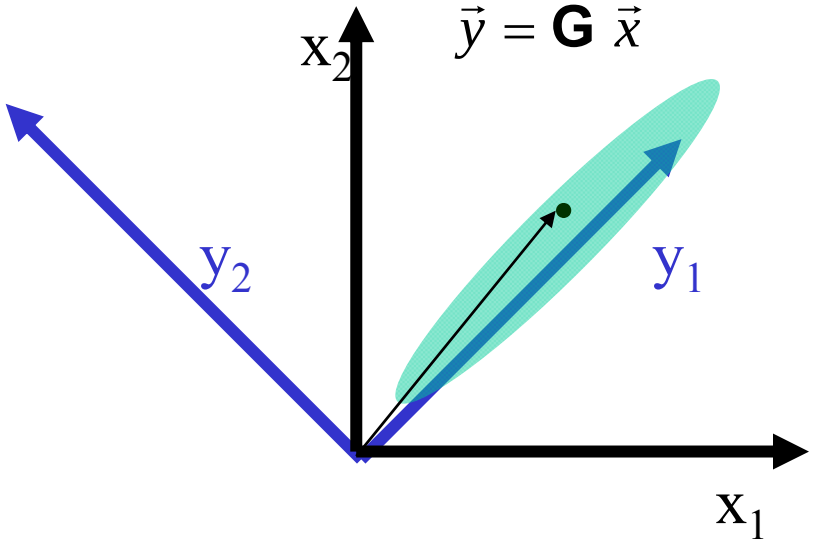
$$\mathbf{\Sigma}_x = \begin{bmatrix} 1.90 & 1.10 \\ 1.10 & 1.10 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \begin{bmatrix} \sigma_{y_1}^2 & 0.00 \\ 0.00 & \sigma_{y_2}^2 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \mathbf{G}\mathbf{\Sigma}_x\mathbf{G}^T$$

Diagonalizing a Matrix

- Presents a clear path of approach:
  - Our input and output covariance matrices must satisfy the conditions



$$\vec{y} = \mathbf{G}\,\vec{x}$$

$$\mathbf{\Sigma}_x = \begin{bmatrix} 1.90 & 1.10 \\ 1.10 & 1.10 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \begin{bmatrix} \sigma_{y_1}^2 & 0.00 \\ 0.00 & \sigma_{y_2}^2 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \mathbf{G}\mathbf{\Sigma}_x\mathbf{G}^T$$

$$\mathbf{G}\mathbf{G}^T = \mathbf{I}$$

$\mathbf{\Sigma}_y$ must be diagonal

G must be Orthogonal

This calls for a solution approach Using *eigenvalues* and *eigenvectors*

What you forgot from Diff Eqs

- Let's create the relationship

$$\vec{a} = \mathbf{M}\,\vec{b}$$

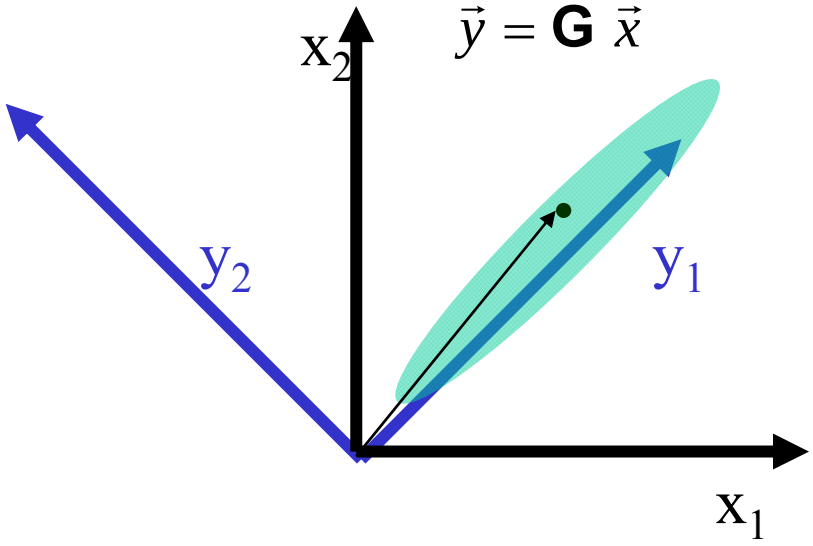
- Let's also envision a scalar parameter λ that likewise accomplishes this for a given vector $\vec{b}$

$$\vec{a} = \mathbf{M}\,\vec{b} = \lambda\,\vec{b}$$

- Through algebra we can say… $\mathbf{M}\,\vec{b} = \lambda\,\vec{b}$

$$\mathbf{M}\,\vec{b} - \lambda\,\vec{b} = 0$$

$$(\mathbf{M} - \lambda\,\mathbf{I})\,\vec{b} = 0$$



- This effectively creates a system of simultaneous equations which we express in linear alg. as

$$\vec{b} = 0 \text{ (trivial solution)}$$

$$\left|\mathbf{M} - \lambda\,\mathbf{I}\right| = 0$$

Presenting a good old polynomial solution for λ.

Eigenvalues

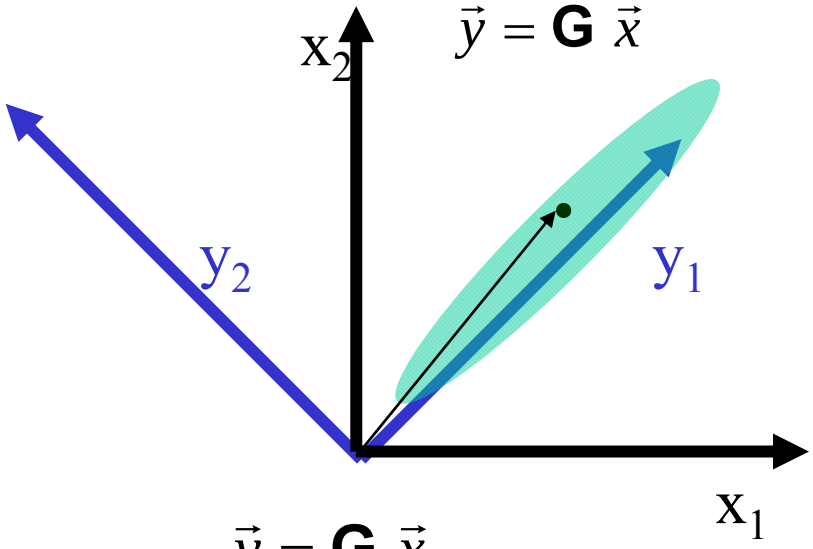
$$\mathbf{M}\vec{b} = \lambda\vec{b}$$

$$\mathbf{M}\vec{b} - \lambda\vec{b} = 0$$

$$(\mathbf{M} - \lambda\mathbf{I})\vec{b} = 0$$

$$|\mathbf{M} - \lambda\mathbf{I}| = 0$$

- These resulting values of λ as we extract them as polynomials are referred to as the "*eigenvalues*" while their corresponding vectors are the "*eigenvectors*"
- What does this have to do with our problem?
  - Let's recall…
  - And…..



$$\vec{y} = \mathbf{G}\,\vec{x}$$

$$\mathbf{\Sigma}_x = \begin{bmatrix} 1.90 & 1.10 \\ 1.10 & 1.10 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \begin{bmatrix} \sigma_{y_1}^2 & 0.00 \\ 0.00 & \sigma_{y_2}^2 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \mathbf{G}\mathbf{\Sigma}_x\mathbf{G}^{\mathsf{T}}$$

$$\mathbf{G}\mathbf{G}^{\mathsf{T}} = \mathbf{I}$$

Eigenvalues

- What does this have to do with our problem?
  - Let's recall…
  - $|\mathbf{M} - \lambda \mathbf{I}| = 0$
  - $|\mathbf{\Sigma}_x - \lambda \mathbf{I}| = 0$
  - So lets do this!

$$\vec{y} = \mathbf{G}\,\vec{x}$$

$$\mathbf{\Sigma}_x = \begin{bmatrix} 1.90 & 1.10 \\ 1.10 & 1.10 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \begin{bmatrix} \sigma_{y_1}^2 & 0.00 \\ 0.00 & \sigma_{y_2}^2 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \mathbf{G}\mathbf{\Sigma}_x\mathbf{G}^T$$

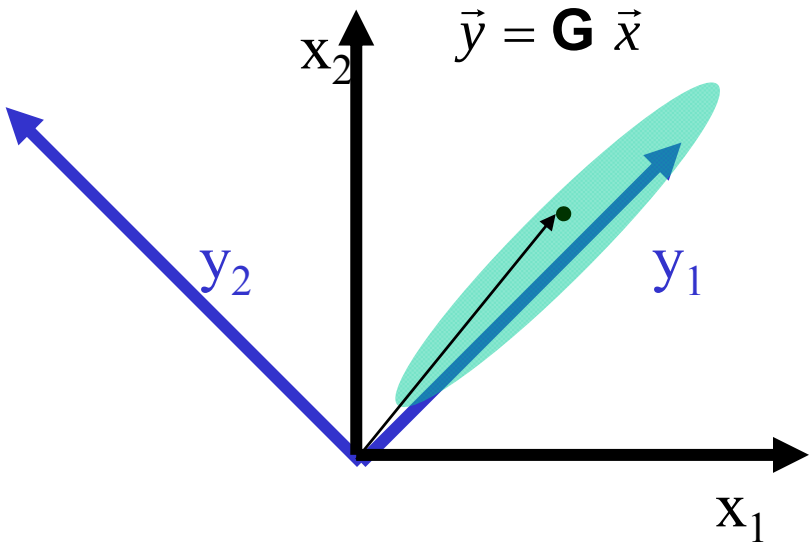
$$\mathbf{G}\mathbf{G}^T = \mathbf{I}$$

$$|\mathbf{\Sigma}_x - \lambda \mathbf{I}| = 0$$

$$\left| \begin{bmatrix} 1.90 - \lambda & 1.10 \\ 1.10 & 1.10 - \lambda \end{bmatrix} \right| = 0$$

$$\lambda^2 - 3\lambda + 0.88 = 0$$

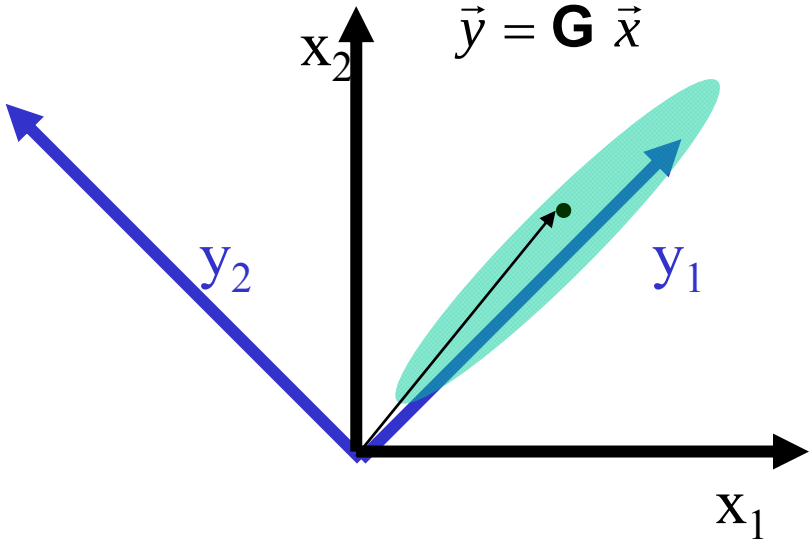
via the quadratic eq, $\lambda = 2.67, 0.33$



- Now to the eigenvectors!

Eigenvectors (and getting "**G**")

- We now have 2 eigenvalues
  - $\lambda = (2.67, 0.33)$
- We now insert them into the full equation from earlier $(\mathbf{\Sigma}_x - \lambda \mathbf{I})\vec{g} = 0$



$$\begin{bmatrix} 1.90 - \lambda & 1.10 \\ 1.10 & 1.10 - \lambda \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = 0$$

$$\vec{g}(2.67) = \begin{bmatrix} 0.82 \\ 0.57 \end{bmatrix}$$

$$\vec{g}(0.33) = \begin{bmatrix} -0.57 \\ 0.82 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} g_{\lambda_1 1} & g_{\lambda_1 2} \\ g_{\lambda_2 1} & g_{\lambda_2 2} \end{bmatrix}$$

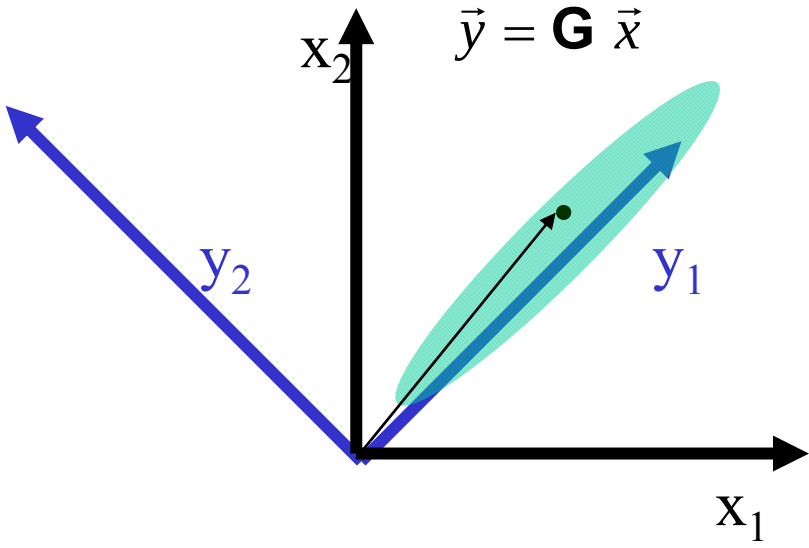
$$(\mathbf{\Sigma}_x - \lambda \mathbf{I})\,\vec{g} = 0$$

$$\mathbf{\Sigma}_y = \mathbf{G}\mathbf{\Sigma}_x\mathbf{G}^{\mathsf{T}}$$

$$\mathbf{G}\mathbf{G}^{\mathsf{T}} = \mathbf{I}$$

Eigenvectors (and getting "**G**")

- Returning to the Eigenvalues and the correlation of our new dataset



$$\lambda = (2.67, 0.33)$$

$$\vec{y} = \mathbf{G}\,\vec{x}$$

$$\mathbf{\Sigma}_x = \begin{bmatrix} 1.90 & 1.10 \\ 1.10 & 1.10 \end{bmatrix}$$

$$\mathbf{\Sigma}_y = \begin{bmatrix} 2.67 & 0.00 \\ 0.00 & 0.33 \end{bmatrix}$$

$$\lambda = (2.67, 0.33)$$

$$\%\text{var} = (89\%, 11\%)$$

$$\mathbf{G} = \begin{bmatrix} g_{\lambda_1 1} & g_{\lambda_1 2} \\ g_{\lambda_2 1} & g_{\lambda_2 2} \end{bmatrix}$$

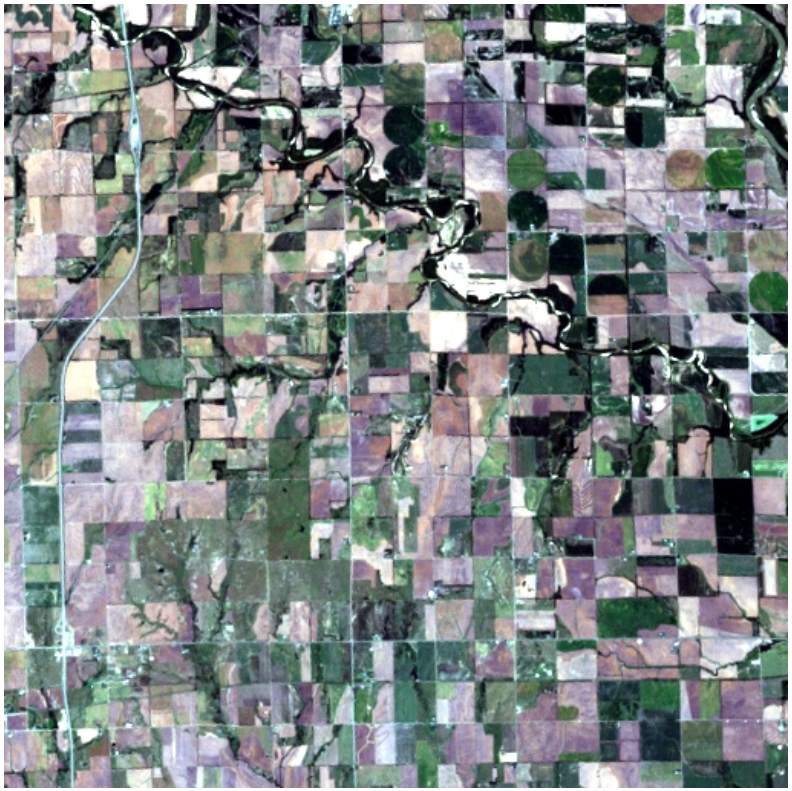
$$(\mathbf{\Sigma}_x - \lambda \mathbf{I})\,\vec{g} = 0$$

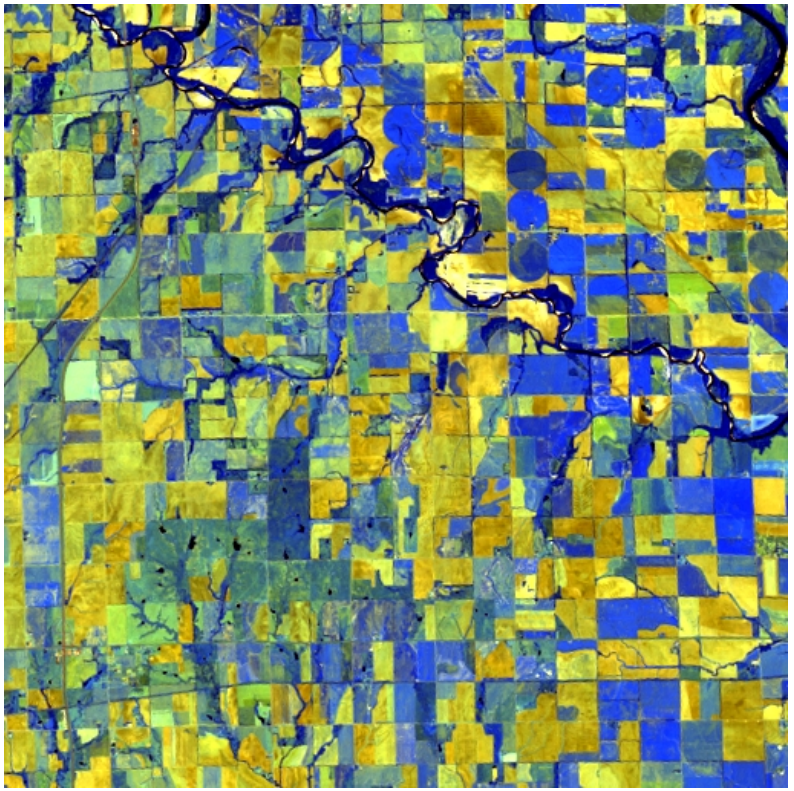
$$\mathbf{\Sigma}_y = \mathbf{G}\mathbf{\Sigma}_x\mathbf{G}^{\mathsf{T}}$$

$$\mathbf{G}\mathbf{G}^{\mathsf{T}} = \mathbf{I}$$

Real Example

- Landsat ETM+ Satellite Imagery (Solar Bands: 12345&7)
  - Walnut Creek Watershed, Kansas.





Visible
Blue: Band 1: 0.479 µm
Green: Band 2: 0.561 µm
Red: Band 3: 0.661 µm

Near Infrared
Blue: Band 4: 0.835 µm
Green: Band 5: 1.650 µm
Red: Band 7: 2.208 µm

Real Example

- Landsat ETM+ Satellite Imagery (Solar Bands: 12345&7)
  - Walnut Creek Watershed, Kansas.

$\Sigma_x =$

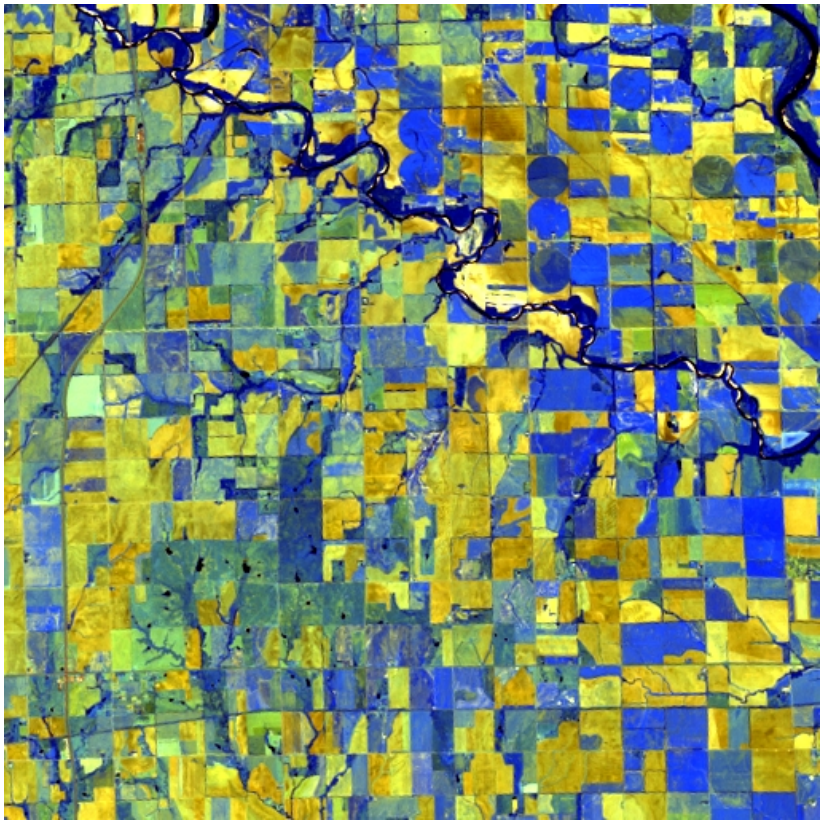
	Band 1	Band 2	Band 3	Band 4	Band 5	Band 6
Band 1	73.62	81.09	171.62	-133.22	189.85	224.16
Band 2	81.09	98.00	196.92	-115.61	215.17	236.79
Band 3	171.62	196.92	443.31	-358.34	483.19	567.82
Band 4	-133.22	-115.61	-358.34	893.62	-315.88	-573.34
Band 5	189.85	215.17	483.19	-315.88	656.76	715.29
Band 7	224.16	236.79	567.82	-573.34	715.29	920.71

$R_x =$

1.0000	0.9547	0.9500	-0.5194	0.8634	0.8610
0.9547	1.0000	0.9448	-0.3907	0.8481	0.7883
0.9500	0.9448	1.0000	-0.5693	0.8955	0.8888
-0.5194	-0.3907	-0.5693	1.0000	-0.4123	-0.6321
0.8634	0.8481	0.8955	-0.4123	1.0000	0.9199
0.8610	0.7883	0.8888	-0.6321	0.9199	1.0000

The Transformed Scene

- But First! Recall what we had before.
  - An image with potentially important information carried across all bands. (Consider the implications when working hyperspectral data with over 200 layers of information!)



Real Example

- The PCA transform and subsequent statistics

$$\lambda = \begin{matrix} 2412.49 \\ 546.44 \\ 86.43 \\ 30.53 \\ 7.85 \\ 2.27 \end{matrix} \qquad G = \begin{matrix} 0.1568 & 0.1676 & 0.4015 & -0.4469 & 0.4736 & 0.6018 \\ -0.0764 & -0.1400 & -0.1605 & -0.8665 & -0.4199 & -0.1471 \\ -0.2638 & -0.4286 & -0.6536 & 0.0491 & 0.1494 & 0.5430 \\ -0.1685 & -0.1368 & -0.1527 & -0.2052 & 0.7554 & -0.5629 \\ -0.6308 & -0.4983 & 0.5857 & 0.0626 & -0.0796 & 0.0215 \\ 0.6883 & -0.7082 & 0.1396 & 0.0315 & 0.0147 & -0.0635 \end{matrix}$$

$$\Sigma_y = \begin{matrix} 2412.49 & -0.00 & 0.00 & -0.00 & -0.00 & 0.00 \\ -0.00 & 546.44 & -0.00 & 0.00 & -0.00 & -0.00 \\ 0.00 & -0.00 & 86.43 & 0.00 & 0.00 & 0.00 \\ -0.00 & 0.00 & 0.00 & 30.53 & -0.00 & -0.00 \\ -0.00 & -0.00 & 0.00 & -0.00 & 7.85 & 0.00 \\ 0.00 & -0.00 & 0.00 & -0.00 & 0.00 & 2.27 \end{matrix}$$

$$R_y = \begin{matrix} 1.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & 0.0000 \\ -0.0000 & 1.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 \\ 0.0000 & -0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.0000 & 0.0000 & 0.0000 & 1.0000 & -0.0000 & -0.0000 \\ -0.0000 & -0.0000 & 0.0000 & -0.0000 & 1.0000 & 0.0000 \\ 0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & 1.0000 \end{matrix}$$

Real Example

- The Implications
  - The layers (or bands – we cannot call them "channels" anymore of you use that jargon) are now optimally uncorrelated from each other.
  - The total relevant information "carried" in the dataset is now compressed to only a few layers.

$$R_x = \begin{bmatrix} 1.0000 & 0.9547 & 0.9500 & -0.5194 & 0.8634 & 0.8610 \\ 0.9547 & 1.0000 & 0.9448 & -0.3907 & 0.8481 & 0.7883 \\ 0.9500 & 0.9448 & 1.0000 & -0.5693 & 0.8955 & 0.8888 \\ -0.5194 & -0.3907 & -0.5693 & 1.0000 & -0.4123 & -0.6321 \\ 0.8634 & 0.8481 & 0.8955 & -0.4123 & 1.0000 & 0.9199 \\ 0.8610 & 0.7883 & 0.8888 & -0.6321 & 0.9199 & 1.0000 \end{bmatrix}$$

$$\Sigma_y = \begin{bmatrix} 2412.49 & -0.00 & 0.00 & -0.00 & -0.00 & 0.00 \\ -0.00 & 546.44 & -0.00 & 0.00 & -0.00 & -0.00 \\ 0.00 & -0.00 & 86.43 & 0.00 & 0.00 & 0.00 \\ -0.00 & 0.00 & 0.00 & 30.53 & -0.00 & -0.00 \\ -0.00 & -0.00 & 0.00 & -0.00 & 7.85 & 0.00 \\ 0.00 & -0.00 & 0.00 & -0.00 & 0.00 & 2.27 \end{bmatrix}$$

$$R_y = \begin{bmatrix} 1.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & 0.0000 \\ -0.0000 & 1.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 \\ 0.0000 & -0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.0000 & 0.0000 & 0.0000 & 1.0000 & -0.0000 & -0.0000 \\ -0.0000 & -0.0000 & 0.0000 & -0.0000 & 1.0000 & 0.0000 \\ 0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

The Transformed Scene: PC_1: λ=2412

- Moving through the PCA layers.
  - Notice how the detail changes as we move through the layers.
  - Recall also that the layer's Eigenvalue is the also the layer's variance!
  - PC_1 is derived from the first row of the transform, **G**.
  - This layer also carries 78% if the image's information.
  - This PC often is a total additive layer akin to "brightness" when using Landsat data *Notice that this is something of an exception! See next slide for more… Hint: Notice that the Near Infrared Band is contrasted against the Red Band!*

$$PC_1 = 0.1568 \cdot TM_1 + 0.1676 \cdot TM_2 + 0.4015 \cdot TM_3 - 0.4469 \cdot TM_4 + 0.4736 \cdot TM_5 + 0.6018 \cdot TM_7$$

$$\frac{\lambda_i}{\sum \lambda} = \frac{\lambda_i}{2412+546+86+31+8+2=3085}$$

The Transformed Scene: PC_2: λ=546

- Moving through the PCA layers.
  - PC_2 is derived from the second row of the transform, **G**.
  - This layer carries 18% if the image's information.
  - *Continuing from the last slide: Notice that THIS layer represents brightness more than the previous one. As indicated earlier this is a bit of an exception to the rule – expect more of these as you apply this approach in your remote sensing research.*
  - *Notice also that a PC layer can be totally negative. But it conveys the same information.*

$$PC_2 = -0.0764 \cdot TM_1 - 0.1400 \cdot TM_2 - 0.1605 \cdot TM_3 - 0.8665 \cdot TM_4 - 0.4199 \cdot TM_5 - 0.1471 \cdot TM_7$$

$$\frac{\lambda_i}{\sum \lambda} = \frac{\lambda_i}{2412+546+86+31+8+2=3085}$$

The Transformed Scene: PC_3: λ=86

- Moving through the PCA layers.
  - PC_3 is derived from the third row of the transform, **G**.
  - This layer carries 3% if the image's information.
  - Notice how the progressive PC's show less and less detail? This is very common.

The Transformed Scene: PC_4: λ=31

- Moving through the PCA layers.
  - PC_4 is derived from the fourth row of the transform, **G**.
  - This layer carries 1% if the image's information.
  - And the usability of the layer degrades further.
  - We are approaching the point where the variance becomes so small that it rivals the precision of the data.

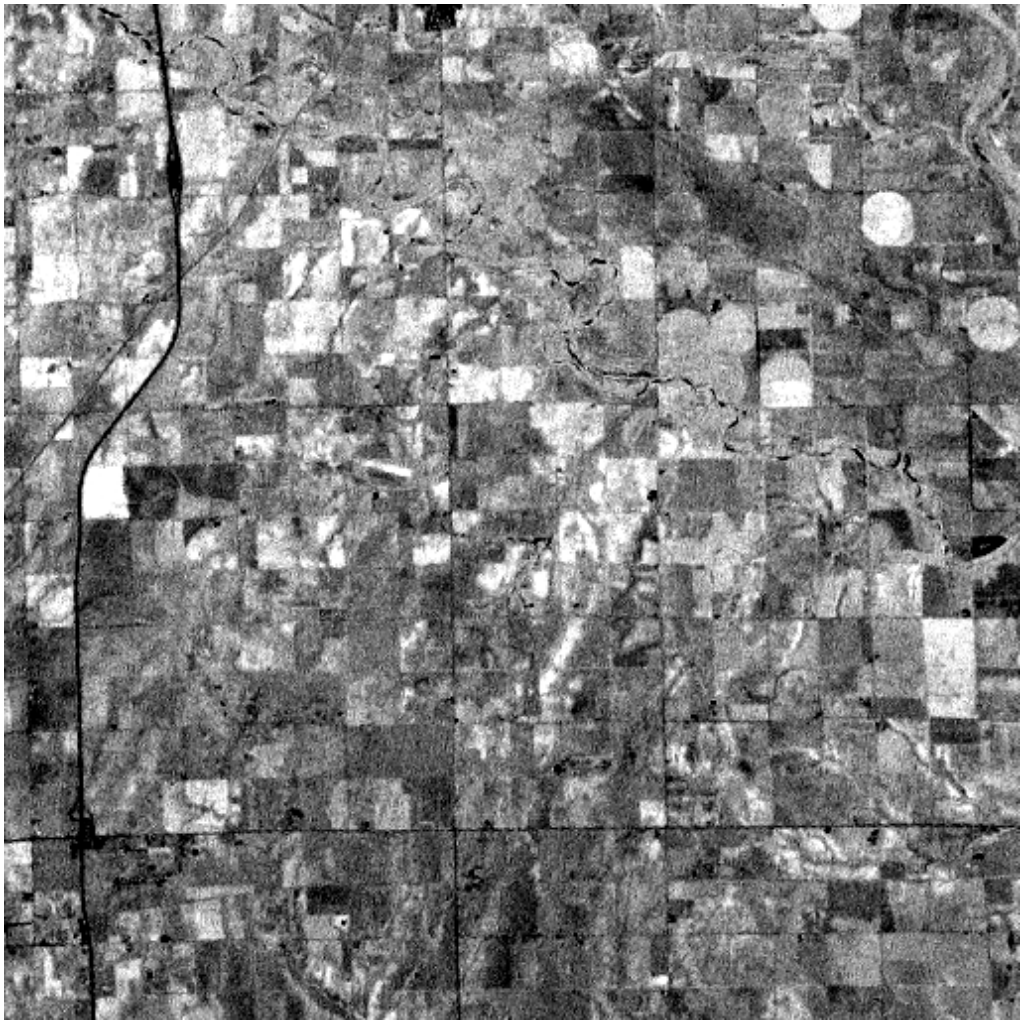
The Transformed Scene: PC_5: λ=8

- Moving through the PCA layers.
  - PC_5 is derived from the fifth row of the transform, **G**.
  - This layer carries 0.3% if the image's information.
  - This image is actually quite noisy.
  - These gifs shown here illustrate the full range of the layer stretched from their minimum to maximum value.

Max-Min Stretch

The Transformed Scene: PC_5: λ=8

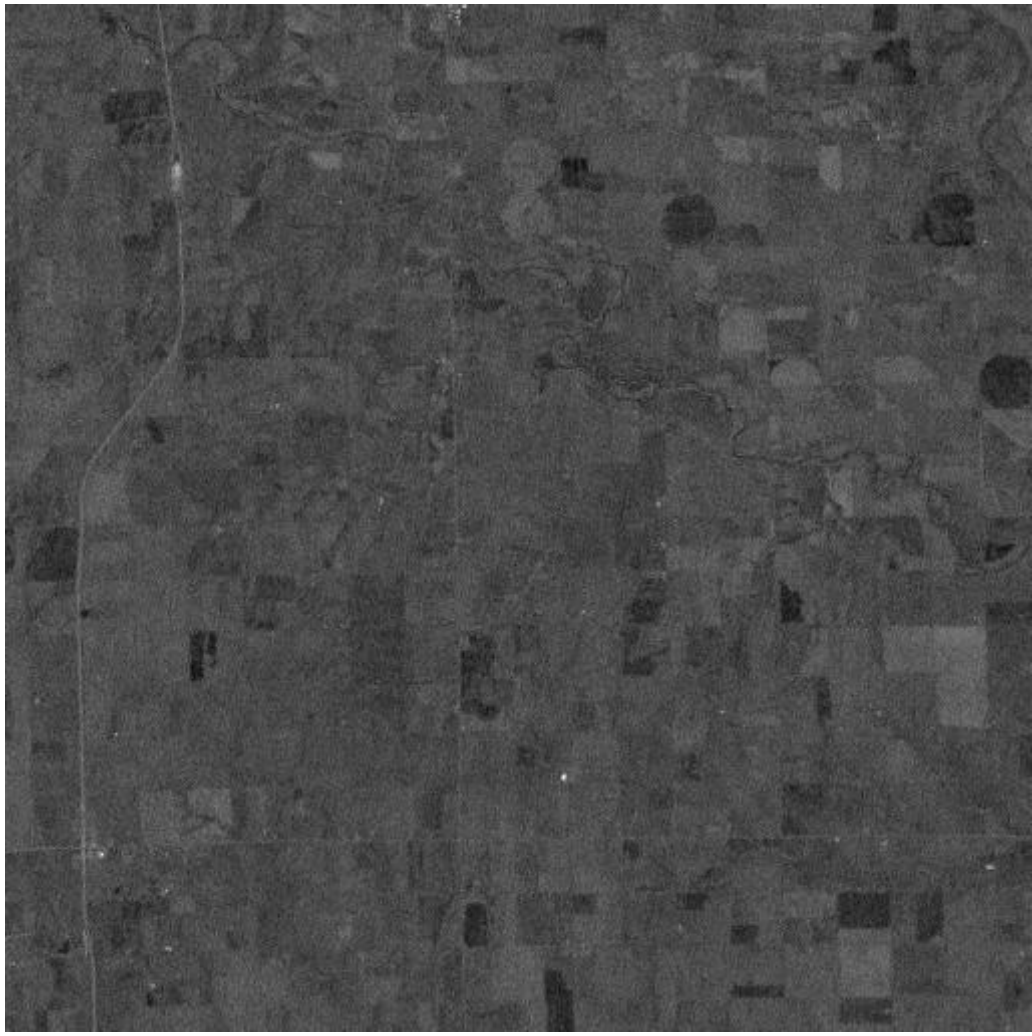
- This scene is the same layer (PC_5) but stretched between the layer values at the 2nd and 98th percentiles. This is a common stretch in image processing: the "2% Stretch."
- This also demonstrates a potential application: Noise Filtering.
  - A clever variant of this approach is used in the MNF transform for hyperspectral data.



2% Stretch

The Transformed Scene: PC_6: λ=2

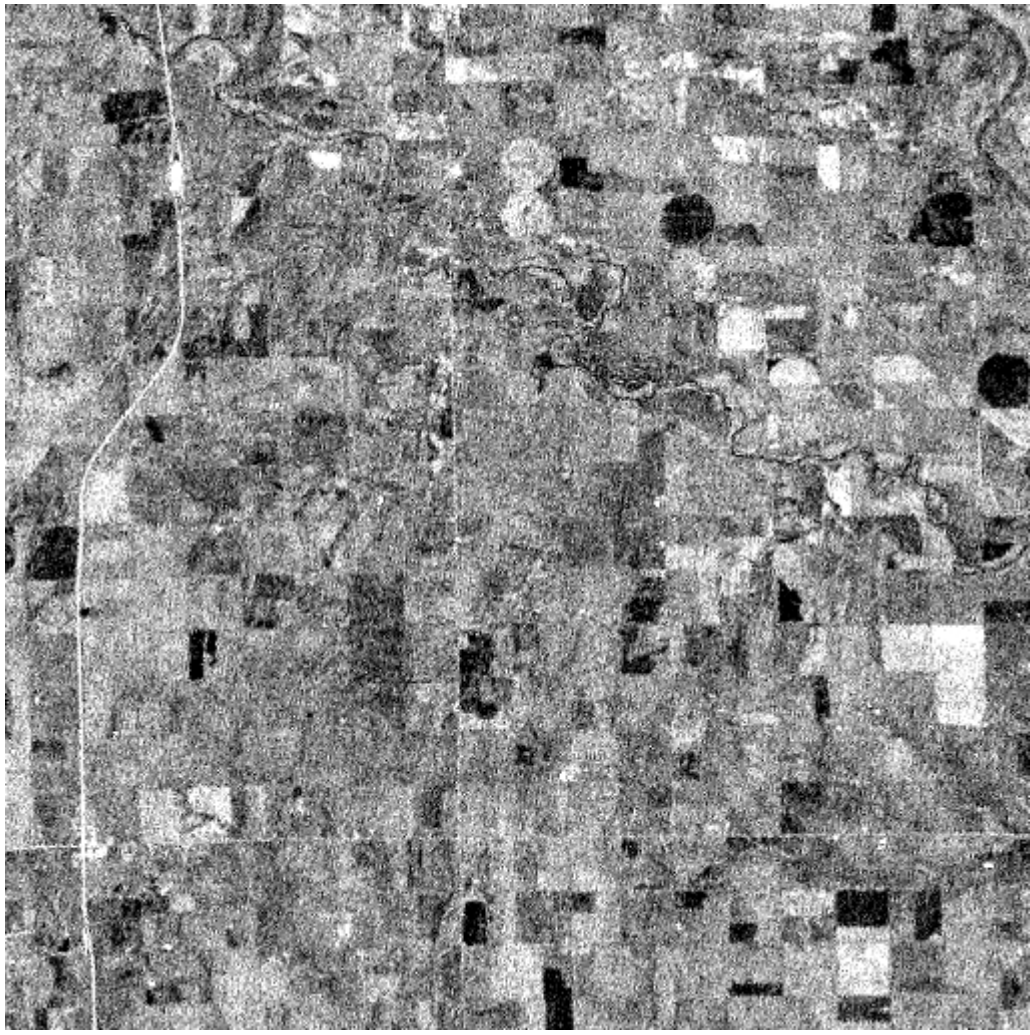
- Moving through the PCA layers.
  - PC_6 is derived from the last row of the transform, **G**.
  - This layer carries 0.06% if the image's information
  - And at this point "if you listen closely, you can hear the bottom of the barrel being scraped."
  - Compared to the informational clarity of the first two principle components, this band is often dismissed as effectively useless.



Max-Min Stretch

The Transformed Scene: PC_6: λ=2

- Moving through the PCA layers.
  - PC_6 is derived from the last row of the transform, **G**.
  - This layer carries 0.06% if the image's information
  - And at this point "if you listen closely, you can hear the bottom of the barrel being scraped."
  - Compared to the informational clarity of the first two principle components, this band is often dismissed as effectively useless.



2% Stretch

Other uses of PCAs

- This method is not limited to use in remote sensing or image processing
- PCAs are very powerful tools when dealing with large multivariate data sets when you are trying to reduce the number of "variable" so that you can handle your datasets more effectively.

Got Questions

- Talk to
  - Dr. Capehart: Remote Sensing